Unipept 6.0: Expanding Metaproteomics Analysis with Support for Missed Cleavages, Semi-Tryptic and Non-Tryptic Peptides

Authors

Tibo Vande Moortele (Tibo.VandeMoortele@UGent.be) (0000-0002-7813-934X)
- Department of Applied Mathematics, Computer Science and Statistics, Ghent University, Ghent, Belgium

Bram Devlaminck (bram.devlaminck@gmail.com) (0009-0009-7656-5458)
- Department of Applied Mathematics, Computer Science and Statistics, Ghent University, Ghent, Belgium

Simon Van de Vyver (Simon.VandeVyver@UGent.be) (0009-0009-7810-2174)
- Department of Applied Mathematics, Computer Science and Statistics, Ghent University, Ghent, Belgium

Tim Van Den Bossche (Tim.VanDenBossche@UGent.be) (0000-0002-5916-2587)
- VIB - UGent Center for Medical Biotechnology, VIB, Ghent, Belgium
- Department of Biomolecular Medicine, Faculty of Medicine and Health Sciences, Ghent University, Ghent, Belgium

Lennart Martens (Lennart.Martens@UGent.be) (0000-0003-4277-658X)
- VIB - UGent Center for Medical Biotechnology, VIB, Ghent, Belgium
- Department of Biomolecular Medicine, Faculty of Medicine and Health Sciences, Ghent University, Ghent, Belgium

Peter Dawyndt (Peter.Dawyndt@UGent.be) (0000-0002-1623-9070)
- Department of Applied Mathematics, Computer Science and Statistics, Ghent University, Ghent, Belgium

Bart Mesuere (Bart.Mesuere@UGent.be) (0000-0003-0610-3441)
- Department of Applied Mathematics, Computer Science and Statistics, Ghent University, Ghent, Belgium

Pieter Verschaffelt (Pieter.Verschaffelt@UGent.be) (0000-0002-6675-1048)
- Department of Applied Mathematics, Computer Science and Statistics, Ghent University, Ghent, Belgium
- VIB - UGent Center for Medical Biotechnology, VIB, Ghent, Belgium
- Department of Biomolecular Medicine, Faculty of Medicine and Health Sciences, Ghent University, Ghent, Belgium

Abstract

Unipept, a pioneering software tool in metaproteomics, has significantly advanced the analysis of complex ecosystems by facilitating both taxonomic and functional insights from environmental samples. From the onset, Unipept's capabilities focused on tryptic peptides, utilizing the predictability and consistency of trypsin digestion to efficiently construct a protein reference database. However, the evolving landscape of proteomics and emerging fields like immunopeptidomics necessitate a more versatile approach that extends beyond the analysis of tryptic peptides. In this article, we present Unipept 6.0, featuring a novel protein reference database, powered by a Sparse Suffix Array index. This advancement enables the analysis of semi-tryptic peptides, peptides with missed cleavages, and non-tryptic peptides such as those from immunopeptidomics (e.g. MHC- and HLA-peptides). A benchmark study demonstrates that this version of Unipept's performance matches or exceeds that of previous versions, while significantly expanding its application range.

Introduction

Unipept[1] is an established ecosystem of user-friendly tools for the downstream analysis of metaproteomics data, initially launched as a web application in 2012 and expanded over the years to include a REST API[2], a command line interface (CLI)[2], and a desktop application[3]. All tools in the Unipept ecosystem depend on the same underlying API server, which can be either accessed via our public server or self-hosted for customized workflows.

Unipept excels in rapidly annotating a set of tryptic peptides with taxonomic and functional information. It accomplishes this by utilizing a highly optimized relational database. For every tryptic peptide in UniProt[4], Unipept's database stores the precalculated taxonomic lowest common ancestor (LCA)[5] and the linked functional annotations (EC[6], GO[7,8], InterPro[9]). For version 2024_03 of UniProt, this results in taxonomic and functional information on 1.3 billion tryptic peptides with a total database size of 1.875 TB. This requires a server with at least 60 GB of memory to efficiently retrieve results. For example, it only takes 11 seconds to analyze the SIHUMIx[10] sample in the CAMPI benchmark dataset[11] of 24424 tryptic peptides.

First and foremost, in metaproteomics, samples are commonly prepared by adding trypsin to cleave proteins at specific sites[12]. This is why constructing a precomputed index of all unique tryptic peptides from UniProtKB is feasible. From time-to-time, however, trypsin will miss a cleavage site, creating a so-called missed cleavage. By default, Unipept 5.0 assumes there are no missed cleavages present in a peptide, which means these cannot be matched to proteins. To solve this problem, a solution was retro-fitted to handle these missed cleavages. By enabling the "advanced missed cleavage" setting, Unipept 5.0 scans the input peptide for missed cleavage positions, splits the peptides and performs separate lookups for each tryptic fragment. The resulting set of proteins are intersected, which results in a collection of possible protein matches, potentially containing the original peptide. Every protein in this collection is then scanned left-to-right to ensure that the original peptide is present, and not only its separate

tryptic fragments. All these extra steps have a significant impact on performance. Enabling “advanced missed cleavage handling” increases the analysis time from 11 seconds to 6 minutes 33 seconds for processing the SIHUMIx dataset S07. In addition, the API and CLI do not support missed cleavage handling. Reviewing literature citing Unipept and our web server logs both reveal that almost all users enable missed cleavage handling, underscoring the importance of optimizing its performance. Although this feature significantly slows down the analysis, it has been widely adopted for its ability to enhance analysis accuracy.

Secondly, Unipept 5.0 cannot deal with semi-tryptic peptides (i.e. peptides where only one terminus is tryptic). This problem becomes particularly significant when proteolytic activity is less predictable, such as in the complex environment of the gut microbiome[13].

Finally, support for matching arbitrary peptides such as non-tryptic peptides, which are often encountered in emerging fields like immunopeptidomics[14], is currently lacking in Unipept 5.0. Immunopeptidomics involves the analysis of HLA (human leukocyte antigen) peptides that play a crucial role in immune system functioning, as they are involved in presenting fragments of proteins (peptides) to T cells, enabling the immune system to distinguish between self and foreign antigens.

To address the current limitations of Unipept 5.0, a new index structure is required that at least ensures the following five properties are met:

1. For every peptide (tryptic or non-tryptic), all protein matches in UniProtKB should be found.
2. Due to current hardware limitations, the maximum memory usage of the new index should be less than 400GiB, while querying, and less than 1TB during construction of the index.
3. Search performance must remain competitive with that of Unipept’s old database index when dealing with tryptic peptides.
4. The new index should at least facilitate all analyses currently supported by Unipept.
5. The new index should allow treating isoleucine and leucine as equivalent during searches.

In this manuscript, we introduce Unipept 6.0, using a new index based on Sparse Suffix Arrays (SSA). Unipept 6.0 allows matching arbitrary peptides (i.e. also semi-tryptic or non-tryptic peptides) instead of only tryptic peptides. It also eliminates the performance penalty of missed cleavage handling and enables this feature in both the API and CLI, and facilitates the five requirements listed above.

Methods

In the Unipept ecosystem, components such as the Unipept Web application, Unipept Desktop app, CLI and third-party applications submit peptides to a central API for analysis. By making

improvements to this API, all tools in the ecosystem automatically benefit. To support arbitrary peptide matching, the relational database in the Unipept API was replaced by a sparse suffix array (SSA), an index designed for efficient searching across multiple sequences. In addition, we completely rewrote the API itself from Ruby-on-Rails to the low-level, performance oriented programming language Rust to achieve even better performance and memory efficiency.

The API now processes input peptides by querying the SSA index, which matches them against all proteins in the UniProtKB database. Relevant taxonomic and functional annotations are retrieved from an optimized in-memory datastore, also implemented in Rust. The API then compiles this information and returns a list of proteins corresponding to each peptide, along with their associated taxonomic and functional annotations, facilitating users with comprehensive metaproteomics data interpretation, and enabling support for the analysis of arbitrary peptides (see Figure 1). An important difference with all previous Unipept versions is that the taxonomic and functional annotations are calculated on-the-fly instead of relying on precalculated values.

The SSA index stores all suffixes of a given set of strings in a sorted manner, allowing for rapid retrieval of matching subsequences. Each suffix is stored as a 64 bit integer, referring to the start of this suffix in the text. Because of the large number of suffixes that need to be referenced this way, a large amount of memory is required to construct the SSA index. However, to reduce memory usage during querying of the SSA index, we can use a combination of sparseness (where only a subset of the suffix array is retained without impacting accuracy), compression and bit packing. The following sections will detail these methods and their impact on both performance and memory efficiency.

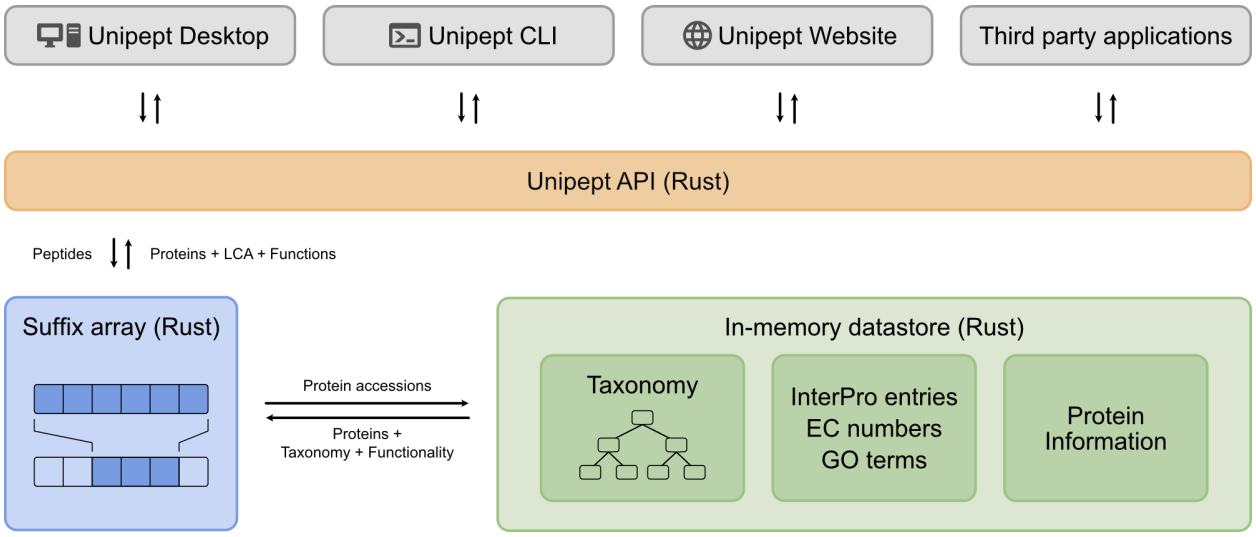


Figure 1: New architecture of Unipept 6.0. Because each component of the Unipept ecosystem (in gray) relies on a common API (in orange) for their data analyses, improving this API immediately enables support for arbitrary peptide analysis and boosts performance for the entire ecosystem.

Constructing the suffix array

The SSA index is built by concatenating all protein sequences in the UniProtKB database. Given the vast number of proteins, this concatenated text and its corresponding suffix array can grow very large. For version 2024_04 of UniProtKB there are almost 246 million proteins, which amounts to 87 billion amino acids in total. Given a 64 bit integer per suffix and the amount of suffixes being equal to the amount of amino acids, this results in a dense suffix array of 64 x 87 x 10^9 bits = 696 GB.

The suffix array itself can be constructed using the popular libdivsufsort[15] or libsais[16] libraries. Both implementations provide linear-time construction algorithms with a $5n + O(1)$ memory complexity, where n is the size of the text being indexed. In our case, constructing the suffix array for version 2024_04 of UniProtKB requires a machine with at least 750 GB of memory and 5 hours of computing time with a single core of a modern CPU (AMD EPYC 7773X) using the libdivsufsort library.

Sparse suffix array

Although a dense suffix array provides optimal performance, it demands a substantial amount of memory. To decrease the amount of memory required for storing the suffix array, a sampling step can be introduced, resulting in a sparse suffix array (SSA). In an SSA, only every k-th suffix of the input text is retained, reducing the array's memory footprint to $1/k^{th}$ of the original.

However, using a sparse suffix array introduces a potential issue: an SSA with sparseness k only allows searching for peptides that are at least k amino acids long, and lookup performance decreases in general. Luckily, this compromise does not introduce a problem for Unipept, since it already limits the search space to peptides that contain 5 amino acids or more, in accordance with mass spectrometer's inability to efficiently detect very short peptides. Additionally, such short peptides are common in many proteins and yield overly generic functional and taxonomic analysis results, which are not very informative.

By using sparseness factor $k = 3$, we significantly reduce the memory requirements of our suffix array while maintaining efficient search capabilities. Starting from our dense suffix array of 696 GB, we get a sparse suffix array requiring only 696 / 3 = 232 GB of memory.

Suffix array compression

Next to increasing the sparseness factor k, we can further decrease the memory requirements by compressing the (sparse) suffix array. Each suffix can be thought of as a position in the text. Because of the vast number of amino acids, by default, the Rust programming language will use 64 bits per suffix. To represent 87 x 10^9 different suffixes, however, we only need $\lceil \log_2(87 * 10^9) \rceil = 37$ bits per suffix, as we don't need the extra bits for larger positions. By storing all suffixes sequentially in a custom bitarray, we save 64 - 37 = 27 bits per suffix. After compressing the sparse suffix array (sparseness factor $k = 3$) of 232 GB, we are left with a compressed sparse suffix array of 133 GB, reducing the required memory footprint by another 57%. This

optimization allows deploying the suffix array on servers with significantly less memory. Figure 2 illustrates the relative memory usage of each component within the final SSA index.

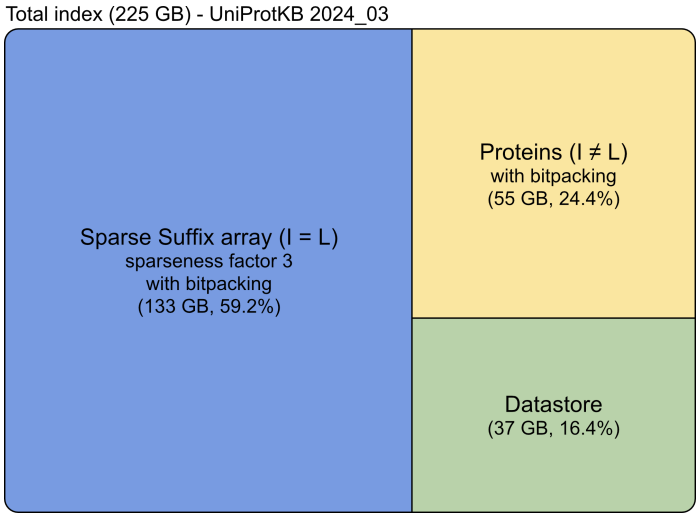


Figure 2: Memory footprint of the individual components in the Unipept 6 index built from UniProtKB version 2024_03.

Equating Isoleucine and Leucine

In metaproteomics data analysis the amino acids isoleucine (I) and leucine (L) are generally treated as equal. The need to treat isoleucine (I) and leucine (L) as equivalent, however, also requires the new index to natively accommodate this functionality. Previous versions of Unipept provided this feature by duplicating the entire database with I and L equated (I = L), essentially maintaining two versions of the index. Applying the same strategy to the suffix array would require twice as much memory, so a different approach is beneficial. Our solution to address this involves two steps. First, rather than construct an SSA for the original protein sequences, we instead index a version where every occurrence of leucine is replaced with isoleucine, ensuring they are treated as identical during the construction of the SSA. Secondly, during the search phase, the same substitution is applied to each peptide. As a result, we find all locations in the original proteins (I ≠ L) for the input peptides where I = L.

Optionally, if users want to perform a search where isoleucine and leucine are considered different characters (I ≠ L), we perform a third filtering step. This way, we filter out incorrect matches from the first step by comparing each occurrence of I and L in the original, unmodified peptide with the corresponding amino acid in the unmodified protein sequences. Any mismatch indicates that I and L were incorrectly equated and thus need to be filtered out, eliminating them from further processing.

Maximal number of matches

A single peptide can potentially match thousands of proteins. Such peptides can cause significant slowdowns during the aggregation phase of an analysis. Moreover, the results for

such peptides are often quite uninformative because these peptides are very common. Out of all tryptic peptides in UniProtKB only 0.001% results in more than 10 000 protein matches. For 95% of those extremely common peptides, the taxonomic lowest common ancestor corresponds to the root node of the NCBI taxonomy (as discussed in the supplementary data S1). Rather than consistently computing the lowest common ancestor, we opted to assign the root to all peptides with more than 10 000 matches. This approach significantly improves performance, with only a minimal reduction in accuracy (just 5% of the 0.001% of all tryptic peptides are assigned a slightly less accurate LCA under this method).

Aggregating peptide annotations

Previous versions of Unipept used a relational database containing all tryptic peptides that can be found in UniProtKB after performing an *in silico* tryptic digest. Because of the limited tryptic search space, it was possible to precompute the lowest common ancestor and aggregate the functional annotations for each tryptic peptide during the construction of the relational database. However, the SSA index needs to compute the taxonomic and functional aggregations on-the-fly. Without restricting the peptide search space to tryptic peptides, there are simply too many possible peptide sequences, making it impossible to apply a precomputation strategy. By reimplementing the aggregation of functional annotations and the computation of the lowest common ancestor in Rust, we achieve a significant performance boost when compared to the original Ruby-on-Rails implementation. This in turn allows us to perform these computations on-the-fly without introducing a significant slowdown.

Results

Replacing the relational database containing precalculated information on all tryptic peptides in UniProtKB by a novel high-performance SSA index with on-the-fly aggregations in Rust enables Unipept 6.0 to analyze arbitrary peptides instead of only tryptic peptides. This added flexibility has a performance penalty that is compensated for by switching to Rust.

To assess the impact of the SSA index on analysis outcomes, we compared the performance of Unipept 5.0 and Unipept 6.0 using sample S07 from the SIHUMIx dataset containing 24424 peptides. In the analysis with Unipept 5.0, 297 peptides (1%) could not be matched. In contrast, Unipept 6.0 successfully matched every peptide in the sample. When examining the taxonomic annotations for each peptide, we observed that 2,680 peptides in Unipept 5.0 were annotated with a Lowest Common Ancestor (LCA) at the species level, whereas this number slightly decreased to 2,551 for Unipept 6.0 (see supplementary data S2).

Further analysis shows that 153 peptides that were assigned a species-level LCA by Unipept 5.0 are matched to higher-rank LCAs when analyzed with Unipept 6.0. This discrepancy is likely due to the expanded number of protein matches in the new SSA index. For Unipept 5.0, when querying a peptide, it was assumed that it was tryptic which in turn contained implicit information about the amino acid occurring before and after the peptide. Unipept 6.0 can now match arbitrary peptides

so this implicit information is no longer used while matching, which can thus result in slightly more matches than before. Conversely, 25 peptides that were unmatched in Unipept 5.0 were successfully assigned a species-level LCA in Unipept 6.0. Upon closer inspection, all of these peptide matches do not align with tryptic peptides in UniProtKB proteins and could therefore not be matched with Unipept 5.0.

Given the minimal differences between the tryptic and non-tryptic indices for tryptic datasets, we have opted not to offer tryptic emulation of the database as an option for Unipept users.

Performance comparison

To assess the performance and accuracy of Unipept 6.0, we used six tryptic peptide samples from the SIHUMIx data set. Each sample contains around 25 000 peptides, including both tryptic peptides and peptides with missed cleavages. Duplicate peptides were not filtered and the “equalize IL” option was enabled. Figure 3 shows wall clock times for searching against Unipept 5.0 with and without missed cleavage handling, and against Unipept 6.0, which natively supports missed cleavage handling.

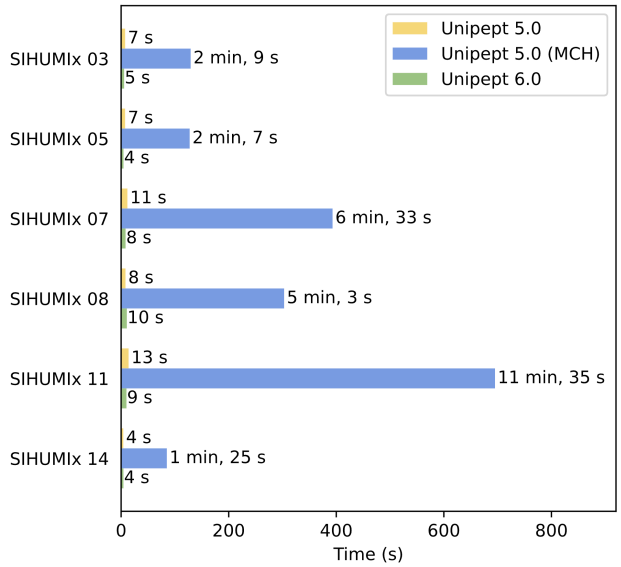


Figure 3: Execution times for Unipept 6.0, Unipept 5.0, and Unipept 5.0 with missed cleavage handling (MCH) enabled. Each SIHUMIx sample contains approximately 25000 tryptic peptides. All analyses used the I = L option.

As a general observation, Unipept 5.0 without missed cleavage handling matches peptides as fast as Unipept 6.0 with missed cleavage handling. This is 20-70 times faster than missed cleavage handling using Unipept 5.0. For CAMPI dataset S07, searches with missed cleavage handling dropped from 6 minutes 33 seconds using Unipept 5.0 to just 8 seconds with Unipept 6.0. This is the same order of magnitude as the 11 seconds it took to analyze the dataset using Unipept 5.0 without missed cleavage handling.

Comparison with other tools

Tools such as the UniProt peptide search tool[17,18], the Expasy ScanProsite tool[19] and Unipept all have the possibility to match peptides against the UniProtKB database. Here, we compare the feature set and performance of these tools to Unipept 6.0. Because of the poor performance of some of these tools, we limited our tests to a single, randomly chosen tryptic peptide *ISPAVLFVIVILAVLFFISGLLHLLVR*.

The feature sets and execution times for all tools are available in Table 1. In terms of available features, the UniProt peptide search tool is identical to Unipept 6.0. They both offer the I = L option and match against the entirety of UniProtKB. However, where the UniProt peptide search tool finishes the search in a few seconds to minutes, Unipept 6.0 requires less than a few milliseconds. The Expasy ScanProsite tool adopts a different approach, offering a broad array of approximate matching options. However, ScanProsite searches only a subset of UniProtKB, limiting its index to proteins from reference genomes. Only proteins part of a reference genome are indexed, which totals to only a third of the complete UniProtKB database. Searching for the reference peptide took 5 minutes using the ScanProsite tool. Unipept 5.0 and Unipept 6.0 both work for the entire UniprotKB, allow equating I and L, and perform equally. Unipept 5.0, however, will only work for tryptic peptides.

	UPS	ESP	Unipept 5.0	Unipept 6.0
Used proteins	all	Ref. proteomes	all	all
Approximate match	[IL]	flexible	[IL]	[IL]
Execution time	1 s - 20 min	5 min	**< 5 ms**	**< 5 ms**
Searchable peptides	all	all	~ tryptic	all

Table 1: Comparison of Unipept 5.0, Unipept 6.0, UniProt peptide search (UPS) tool and ExpasyScanProsite (EPS) tool. [IL] in the approximate matching row indicates only I and L can be equated, while ~ tryptic in the searchable peptides row indicates only tryptic peptides or tryptic peptides with missed cleavages can be found.

Conclusions

In this manuscript, we introduce a new version of Unipept that relies on a sparse suffix array for peptide searches. This index is implemented in Rust and allows fast on-demand calculation of both taxonomic and functional annotations for a set of peptides. Because data processing no longer relies on precalculated results for tryptic peptides, Unipept 6.0 can now match arbitrary peptides. This allows us to perform “missed cleavage handling” by default up to 70 times faster than the time it took with Unipept 5.0. In turn, this enables missed cleavage handling support in the API, CLI and third-party tools which was previously not possible.

The introduction of an SSA index in Unipept 6.0 thus broadens its applicability to emerging fields such as immunopeptidomics, because of its support for matching MHC or HLA peptides.

Looking ahead, Unipept is well-equipped to evolve further and provides a solid foundation for migrating features currently exclusive to the desktop application to the web platform. A notable example of this is the potential introduction of an ad-hoc filter on UniProtKB, comparable to the custom database construction that is already available in the Unipept Desktop app. This feature would enable researchers to more effectively utilize taxonomic data, such as those derived from prior metagenomic experiments, by querying a targeted subset of proteins within the vast UniProtKB database. By migrating these advanced features to the web, we make them more accessible to a broader range of users, improving the overall user experience and ensuring that these tools are available without the need for installing software locally.

Acknowledgements

PV acknowledges funding by Ghent University [BOF/01P10623]. L.M. acknowledges funding from the Research Foundation Flanders (FWO) [G028821N] and [G010023N]. TVDB acknowledges funding by the Research Foundation Flanders (FWO) [1286824N].

References

(1) Gurdeep Singh, R.; Tanca, A.; Palomba, A.; Van der Jeugt, F.; Verschaffelt, P.; Uzzau, S.; Martens, L.; Dawyndt, P.; Mesuere, B. Unipept 4.0: Functional Analysis of Metaproteome Data. *J. Proteome Res.* **2019**, *18* (2), 606–615. https://doi.org/10.1021/acs.jproteome.8b00716.
(2) Verschaffelt, P.; Van Thienen, P.; Van Den Bossche, T.; Van der Jeugt, F.; De Tender, C.; Martens, L.; Dawyndt, P.; Mesuere, B. Unipept CLI 2.0: Adding Support for Visualizations and Functional Annotations. *Bioinformatics* **2020**, *36* (14), 4220–4221. https://doi.org/10.1093/bioinformatics/btaa553.
(3) *Unipept Desktop 2.0: Construction of Targeted Reference Protein Databases for Metaproteogenomics Analyses | Journal of Proteome Research*. https://pubs.acs.org/doi/full/10.1021/acs.jproteome.3c00091 (accessed 2024-09-23).
(4) The UniProt Consortium. UniProt: The Universal Protein Knowledgebase in 2023. *Nucleic Acids Res.* **2023**, *51* (D1), D523–D531. https://doi.org/10.1093/nar/gkac1052.
(5) Van Den Bossche, T.; Verschaffelt, P.; Vande Moortele, T.; Dawyndt, P.; Martens, L.; Mesuere, B. Biodiversity Analysis of Metaproteomics Samples with Unipept: A Comprehensive Tutorial. In *Protein Bioinformatics*; Lisacek, F., Ed.; Springer US: New York, NY, 2024; pp 183–215. https://doi.org/10.1007/978-1-0716-4007-4_11.
(6) Webb, E. Enzyme Nomenclature 1992. Recommendations of the Nomenclature Committee of the International Union of Biochemistry and Molecular Biology on the Nomenclature and Classification of Enzymes.; 1992.
(7) Ashburner, M.; Ball, C. A.; Blake, J. A.; Botstein, D.; Butler, H.; Cherry, J. M.; Davis, A. P.; Dolinski, K.; Dwight, S. S.; Eppig, J. T.; Harris, M. A.; Hill, D. P.; Issel-Tarver, L.; Kasarskis, A.; Lewis, S.; Matese, J. C.; Richardson, J. E.; Ringwald, M.; Rubin, G. M.; Sherlock, G. Gene Ontology: Tool for the Unification of Biology. *Nat. Genet.* **2000**, *25* (1), 25–29. https://doi.org/10.1038/75556.
(8) The Gene Ontology Consortium; Aleksander, S. A.; Balhoff, J.; Carbon, S.; Cherry, J. M.; Drabkin, H. J.; Ebert, D.; Feuermann, M.; Gaudet, P.; Harris, N. L.; Hill, D. P.; Lee, R.; Mi, H.; Moxon, S.; Mungall, C. J.; Muruganugan, A.; Mushayahama, T.; Sternberg, P. W.; Thomas, P. D.; Van Auken, K.; Ramsey, J.; Siegele, D. A.; Chisholm, R. L.; Fey, P.;

Aspromonte, M. C.; Nugnes, M. V.; Quaglia, F.; Tosatto, S.; Giglio, M.; Nadendla, S.; Antonazzo, G.; Attrill, H.; dos Santos, G.; Marygold, S.; Strelets, V.; Tabone, C. J.; Thurmond, J.; Zhou, P.; Ahmed, S. H.; Asanitthong, P.; Luna Buitrago, D.; Erdol, M. N.; Gage, M. C.; Ali Kadhum, M.; Li, K. Y. C.; Long, M.; Michalak, A.; Pesala, A.; Pritazahra, A.; Saverimuttu, S. C. C.; Su, R.; Thurlow, K. E.; Lovering, R. C.; Logie, C.; Oliferenko, S.; Blake, J.; Christie, K.; Corbani, L.; Dolan, M. E.; Drabkin, H. J.; Hill, D. P.; Ni, L.; Sitnikov, D.; Smith, C.; Cuzick, A.; Seager, J.; Cooper, L.; Elser, J.; Jaiswal, P.; Gupta, P.; Jaiswal, P.; Naithani, S.; Lera-Ramirez, M.; Rutherford, K.; Wood, V.; De Pons, J. L.; Dwinell, M. R.; Hayman, G. T.; Kaldunski, M. L.; Kwitek, A. E.; Laulederkind, S. J. F.; Tutaj, M. A.; Vedi, M.; Wang, S.-J.; D'Eustachio, P.; Aimo, L.; Axelsen, K.; Bridge, A.; Hyka-Nouspikel, N.; Morgat, A.; Aleksander, S. A.; Cherry, J. M.; Engel, S. R.; Karra, K.; Miyasato, S. R.; Nash, R. S.; Skrzypek, M. S.; Weng, S.; Wong, E. D.; Bakker, E.; Berardini, T. Z.; Reiser, L.; Auchincloss, A.; Axelsen, K.; Argoud-Puy, G.; Blatter, M.-C.; Boutet, E.; Breuza, L.; Bridge, A.; Casals-Casas, C.; Coudert, E.; Estreicher, A.; Livia Famiglietti, M.; Feuermann, M.; Gos, A.; Gruaz-Gumowski, N.; Hulo, C.; Hyka-Nouspikel, N.; Jungo, F.; Le Mercier, P.; Lieberherr, D.; Masson, P.; Morgat, A.; Pedruzzi, I.; Pourcel, L.; Poux, S.; Rivoire, C.; Sundaram, S.; Bateman, A.; Bowler-Barnett, E.; Bye-A-Jee, H.; Denny, P.; Ignatchenko, A.; Ishtiaq, R.; Lock, A.; Lussi, Y.; Magrane, M.; Martin, M. J.; Orchard, S.; Raposo, P.; Speretta, E.; Tyagi, N.; Warner, K.; Zaru, R.; Diehl, A. D.; Lee, R.; Chan, J.; Diamantakis, S.; Raciti, D.; Zarowiecki, M.; Fisher, M.; James-Zorn, C.; Ponferrada, V.; Zorn, A.; Ramachandran, S.; Ruzicka, L.; Westerfield, M. The Gene Ontology Knowledgebase in 2023. *Genetics* **2023**, *224* (1), iyad031. https://doi.org/10.1093/genetics/iyad031.

(9) Paysan-Lafosse, T.; Blum, M.; Chuguransky, S.; Grego, T.; Pinto, B. L.; Salazar, G. A.; Bileschi, M. L.; Bork, P.; Bridge, A.; Colwell, L.; Gough, J.; Haft, D. H.; Letunić, I.; Marchler-Bauer, A.; Mi, H.; Natale, D. A.; Orengo, C. A.; Pandurangan, A. P.; Rivoire, C.; Sigrist, C. J. A.; Sillitoe, I.; Thanki, N.; Thomas, P. D.; Tosatto, S. C. E.; Wu, C. H.; Bateman, A. InterPro in 2022. *Nucleic Acids Res.* **2023**, *51* (D1), D418–D427. https://doi.org/10.1093/nar/gkac993.

(10) Krause, J. L.; Schaepe, S. S.; Fritz-Wallace, K.; Engelmann, B.; Rolle-Kampczyk, U.; Kleinsteuber, S.; Schattenberg, F.; Liu, Z.; Mueller, S.; Jehmlich, N.; Von Bergen, M.; Herberth, G. Following the Community Development of SIHUMIx – a New Intestinal in Vitro Model for Bioreactor Use. *Gut Microbes* **2020**, *11* (4), 1116–1129. https://doi.org/10.1080/19490976.2019.1702431.

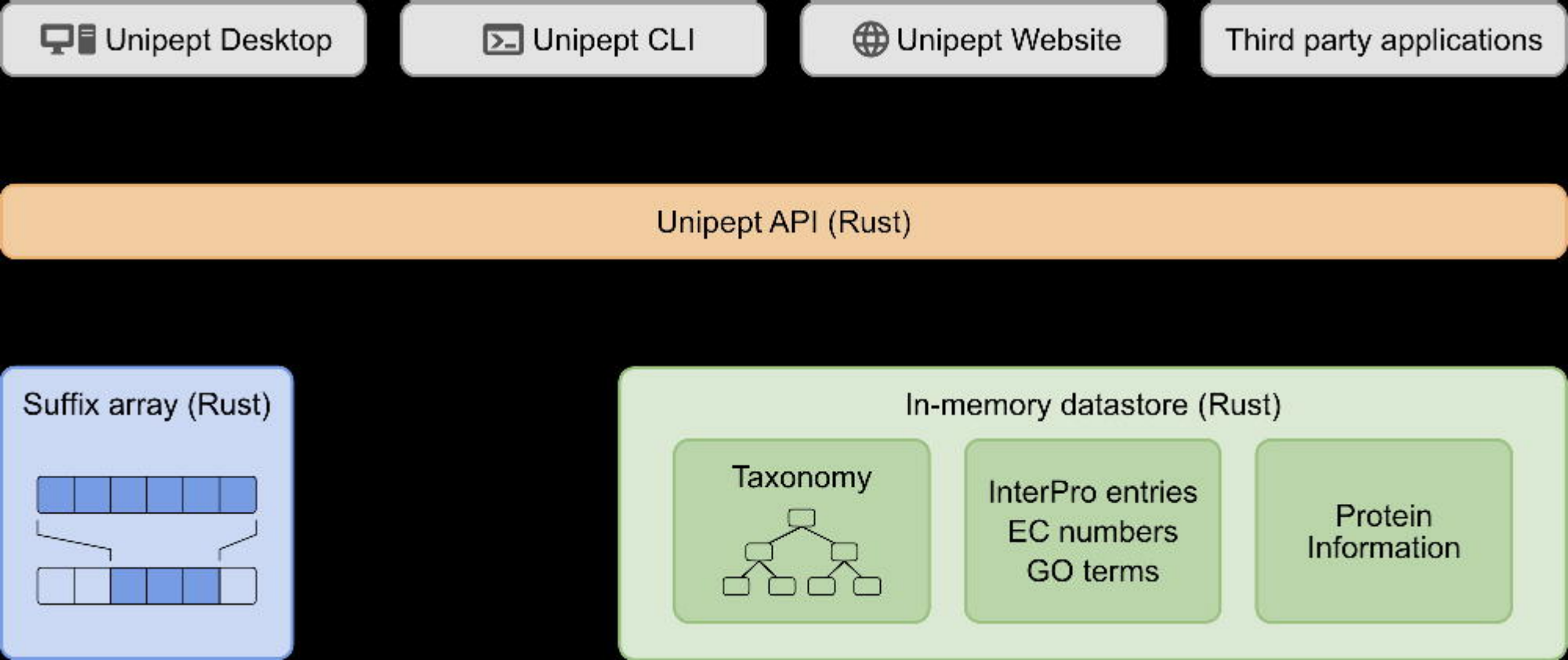
(11) Van Den Bossche, T.; Kunath, B. J.; Schallert, K.; Schäpe, S. S.; Abraham, P. E.; Armengaud, J.; Arntzen, M. Ø.; Bassignani, A.; Benndorf, D.; Fuchs, S.; Giannone, R. J.; Griffin, T. J.; Hagen, L. H.; Halder, R.; Henry, C.; Hettich, R. L.; Heyer, R.; Jagtap, P.; Jehmlich, N.; Jensen, M.; Juste, C.; Kleiner, M.; Langella, O.; Lehmann, T.; Leith, E.; May, P.; Mesuere, B.; Miotello, G.; Peters, S. L.; Pible, O.; Queiros, P. T.; Reichl, U.; Renard, B. Y.; Schiebenhoefer, H.; Sczyrba, A.; Tanca, A.; Trappe, K.; Trezzi, J.-P.; Uzzau, S.; Verschaffelt, P.; von Bergen, M.; Wilmes, P.; Wolf, M.; Martens, L.; Muth, T. Critical Assessment of MetaProteome Investigation (CAMPI): A Multi-Laboratory Comparison of Established Workflows. *Nat. Commun.* **2021**, *12* (1), 7305. https://doi.org/10.1038/s41467-021-27542-8.

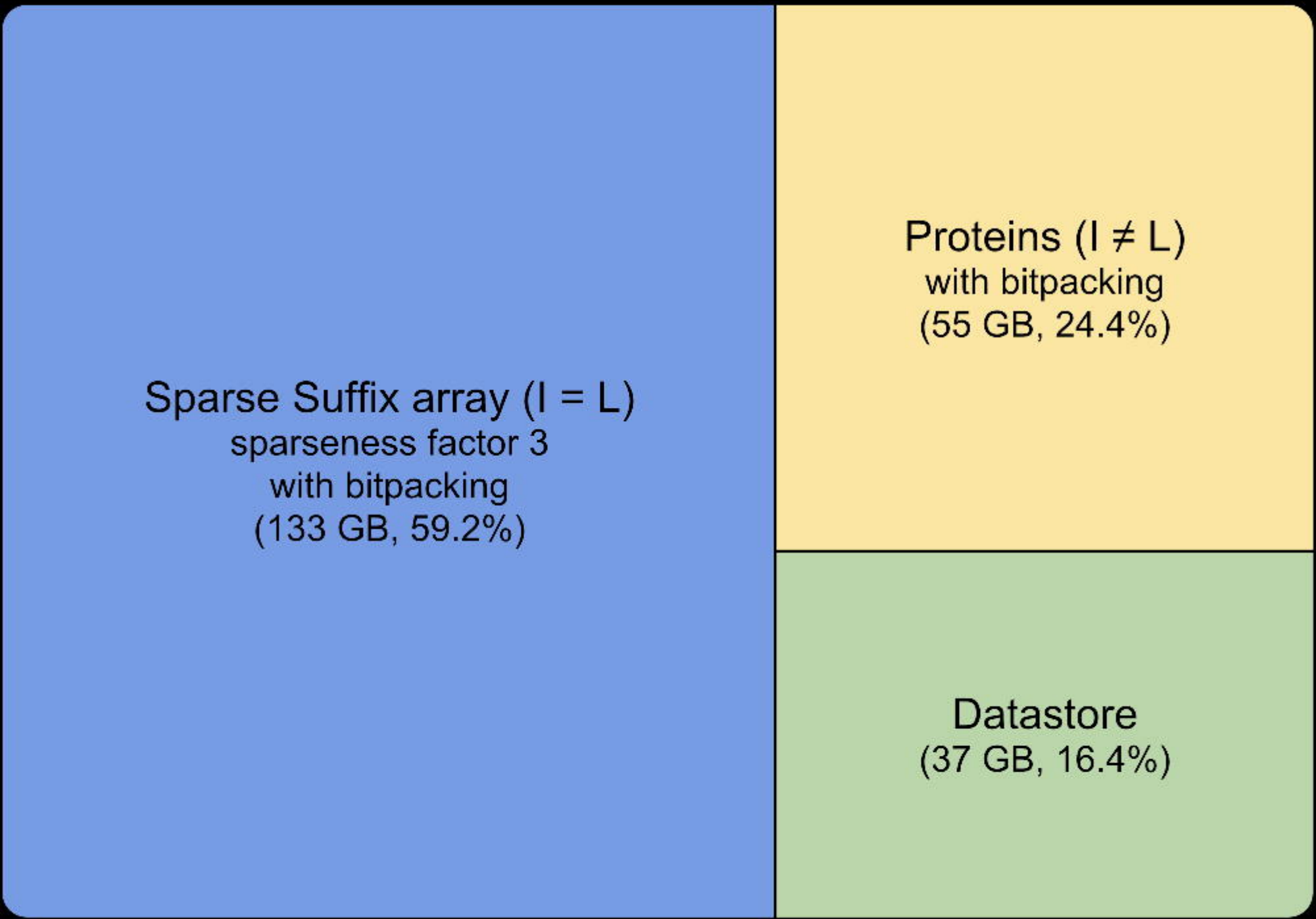
(12) Olsen, J. V.; Ong, S.-E.; Mann, M. Trypsin Cleaves Exclusively C-Terminal to Arginine and Lysine Residues *. *Mol. Cell. Proteomics* **2004**, *3* (6), 608–614. https://doi.org/10.1074/mcp.T400003-MCP200.

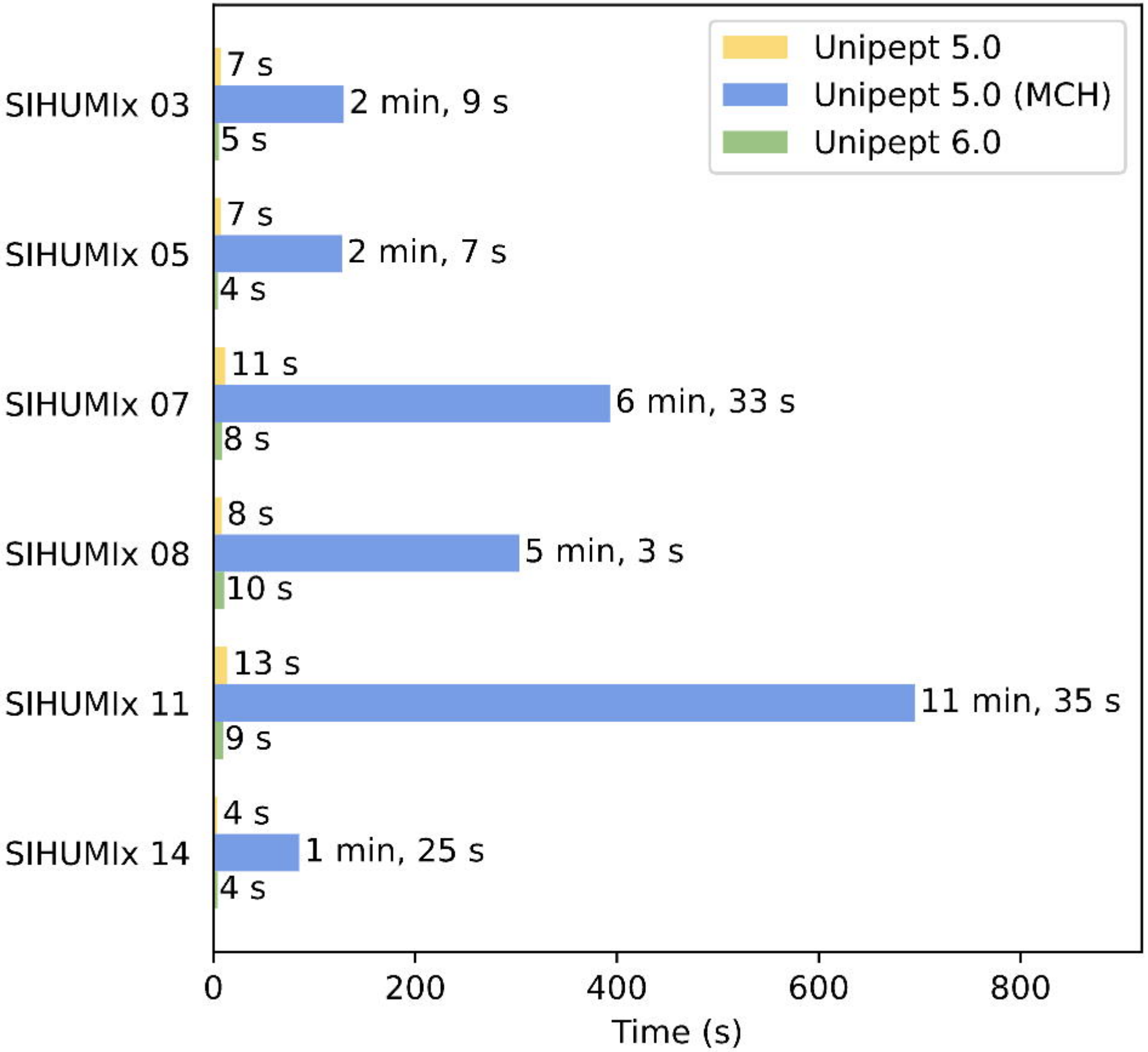
(13) Yan, Z.; He, F.; Xiao, F.; He, H.; Li, D.; Cong, L.; Lin, L.; Zhu, H.; Wu, Y.; Yan, R.; Li, X.; Shan, H. A Semi-Tryptic Peptide Centric Metaproteomic Mining Approach and Its Potential Utility in Capturing Signatures of Gut Microbial Proteolysis. *Microbiome* **2021**, *9* (1), 12. https://doi.org/10.1186/s40168-020-00967-x.

(14) Shapiro, I. E.; Bassani-Sternberg, M. The Impact of Immunopeptidomics: From Basic

Research to Clinical Implementation. *Semin. Immunol.* **2023**, *66*, 101727. https://doi.org/10.1016/j.smim.2023.101727.
(15) Mori, Y. Y-256/Libdivsufsort, 2024. https://github.com/y-256/libdivsufsort (accessed 2024-09-17).
(16) Grebnov, I. IlyaGrebnov/Libsais, 2024. https://github.com/IlyaGrebnov/libsais (accessed 2024-09-17).
(17) *Peptide search | UniProt*. https://www.uniprot.org/peptide-search (accessed 2024-09-24).
(18) Chen, C.; Li, Z.; Huang, H.; Suzek, B. E.; Wu, C. H.; UniProt Consortium. A Fast Peptide Match Service for UniProt Knowledgebase. *Bioinforma. Oxf. Engl.* **2013**, *29* (21), 2808–2809. https://doi.org/10.1093/bioinformatics/btt484.
(19) *ScanProsite*. https://prosite.expasy.org/scanprosite/ (accessed 2024-09-24).


Unipept Desktop
Unipept CLI
Unipept Website
Third party applications
Unipept API (Rust)
Suffix array (Rust)
In-memory datastore (Rust)
Taxonomy
InterPro entries
EC numbers
GO terms
Protein
Information


Sparse Suffix array (I = L)
sparseness factor 3
with bitpacking
(133 GB, 59.2%)
Proteins (I ≠ L)
with bitpacking
(55 GB, 24.4%)
Datastore
(37 GB, 16.4%)


Unipept 5.0
Unipept 5.0 (MCH)
Unipept 6.0
SIHUMIx 03
7 s
2 min, 9 s
5 s
SIHUMIx 05
7 s
2 min, 7 s
4 s
SIHUMIx 07
11 s
6 min, 33 s
8 s
SIHUMIx 08
8 s
5 min, 3 s
10 s
SIHUMIx 11
13 s
11 min, 35 s
9 s
SIHUMIx 14
4 s
1 min, 25 s
4 s
0
200
400
600
800
Time (s)